

REVERSE ENGINEERING WEB 2.0 APPLICATIONS



© Blueinfy Solutions Pvt. Ltd.

Blueinfy

Who Am I?



<http://shreeraj.blogspot.com>
shreeraj@blueinfy.com
<http://www.blueinfy.com>

- **Founder & Director**

- Blueinfy Solutions Pvt. Ltd. (Brief)
- SecurityExposure.com

- **Past experience**

- Net Square, Chase, IBM & Foundstone

- **Interest**

- Web security research

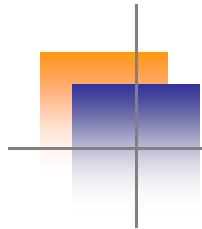
- **Published research**

- Articles / Papers – Securityfocus, O'erilly, DevX, InformIT etc.
- Tools – wsScanner, scanweb2.0, AppMap, AppCodeScan, AppPrint etc.
- Advisories - .Net, Java servers etc.

- **Books (Author)**

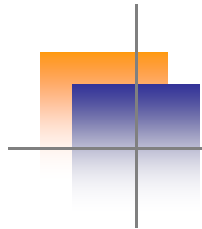
- Web 2.0 Security – Defending Ajax, RIA and SOA
- Hacking Web Services
- Web Hacking





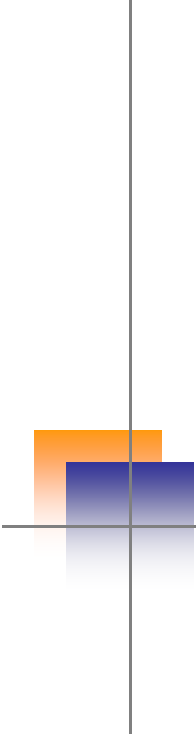
Real Life Cases

- Large Portal and Web 2.0 Trading App
- Scanner failed
 - Reason : Flash and Ajax application
 - Couldn't crawl at first place, what to scan?
- Manual testing with some scripts
 - DOM based crawling
 - RIA reverse engineering
 - Collecting assets from the application

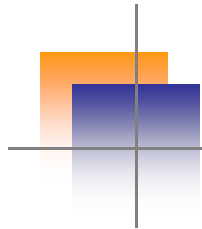


Real Life Cases

- Findings & Vulnerabilities
 - Client side business logic bypass
 - DOM based XSS
 - Poor Ajax routines
 - AMF with SQL injections
 - Vulnerable flash files
 - Various other injections – XPATH and LDAP
 - Impact : Financial loss and identity theft ...

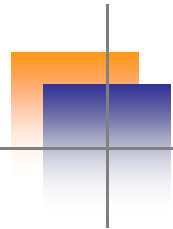


Trends, Attacks & Architecture Web 2.0

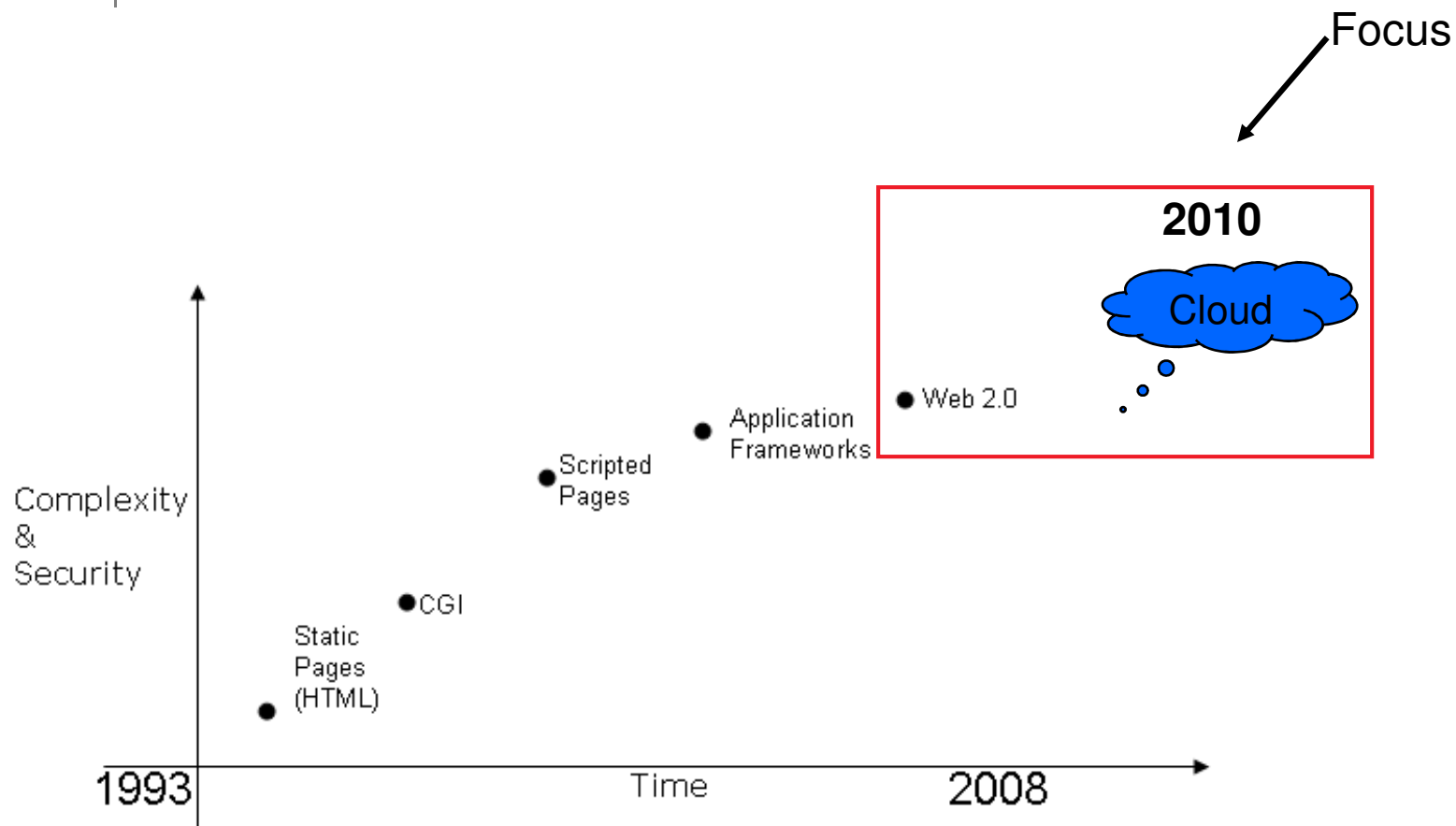


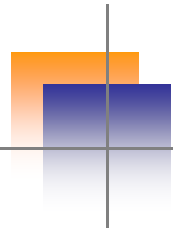
Technology Trends

- Web 2.0 – Ajax, Silverlight and Flex/Flash
- Web Services and SOA
- Cloud APIs and SaaS
- Browser empowering – HTML 5 and several other features
- Traditional stacks are evolving around frameworks



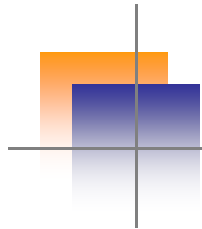
Past, Present and Future





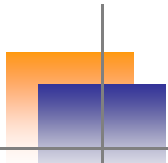
Web Attacks and Targets

- 80% Sites are having security issues
- Web Application Layer vulnerabilities are growing at higher rate in security space
- Client side hacking and vulnerabilities are on the rise – from 5% to 30% (IBM)
- Web browser vulnerabilities is growing at high rate
- End point exploitation shifting from OS to browser and its plugins

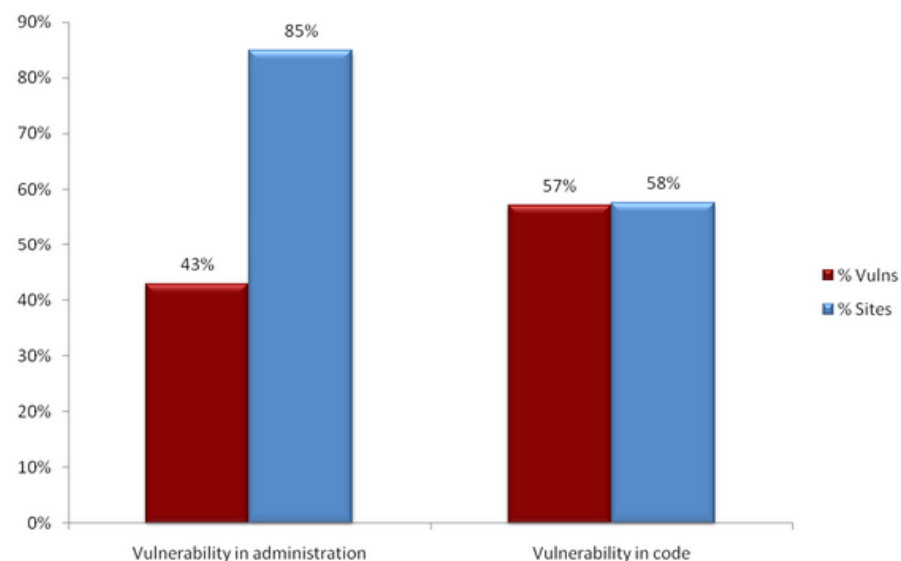
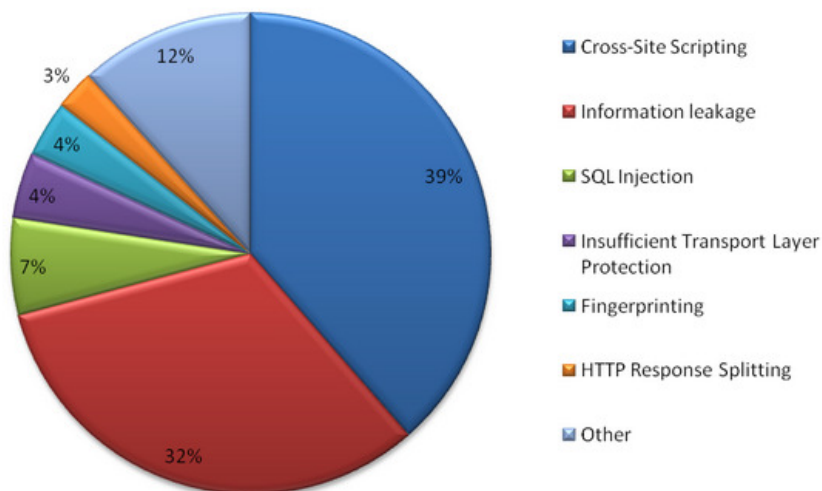
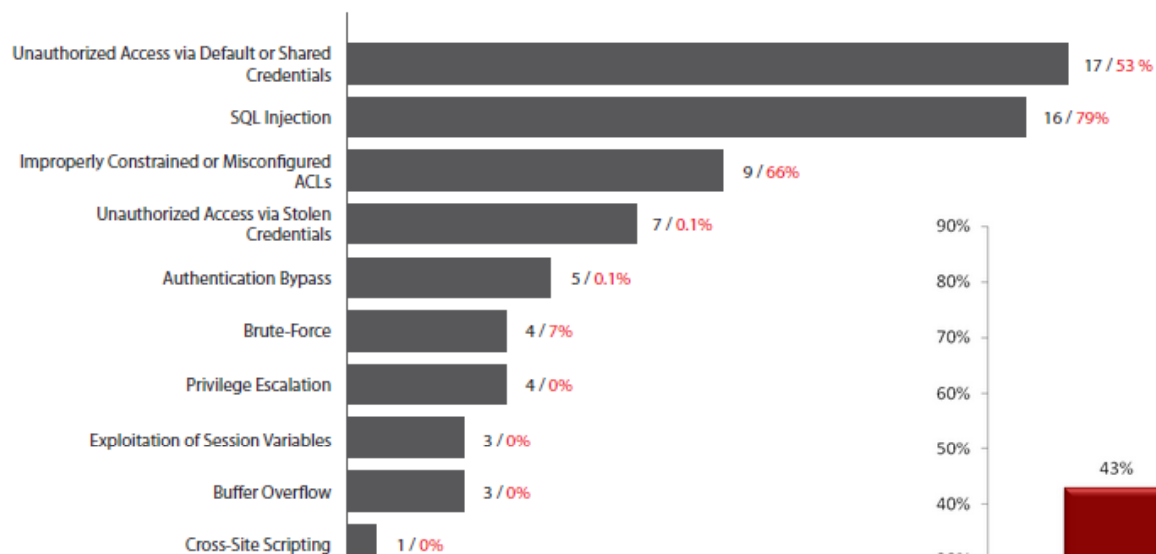


Web 2.0 Attacks

- **Cross Site Scripting (XSS)**
 - Web 2.0 systems such as social networks, blogs or wikis, making Web 2.0 applications especially vulnerable to XSS.
 - DOM based XSS
 - Ajax and Flash issues
- **Cross Site Request Forgery (CSRF) / Cross Gadget Request Forgery (CGRF)-**
- **Phishing** – DOM based redirects
- **Information Leakage** – client side leaks and errors/exceptions
- **Injection Flaws** – XPATH, LDAP, JavaScript & JSON
- **Information Integrity** – mashups and un-trusted sources
- **Insufficient Anti-Automation** – CSRF and Phishing



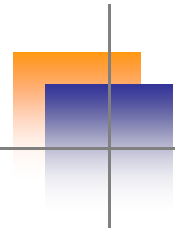
Hacks & Attacks



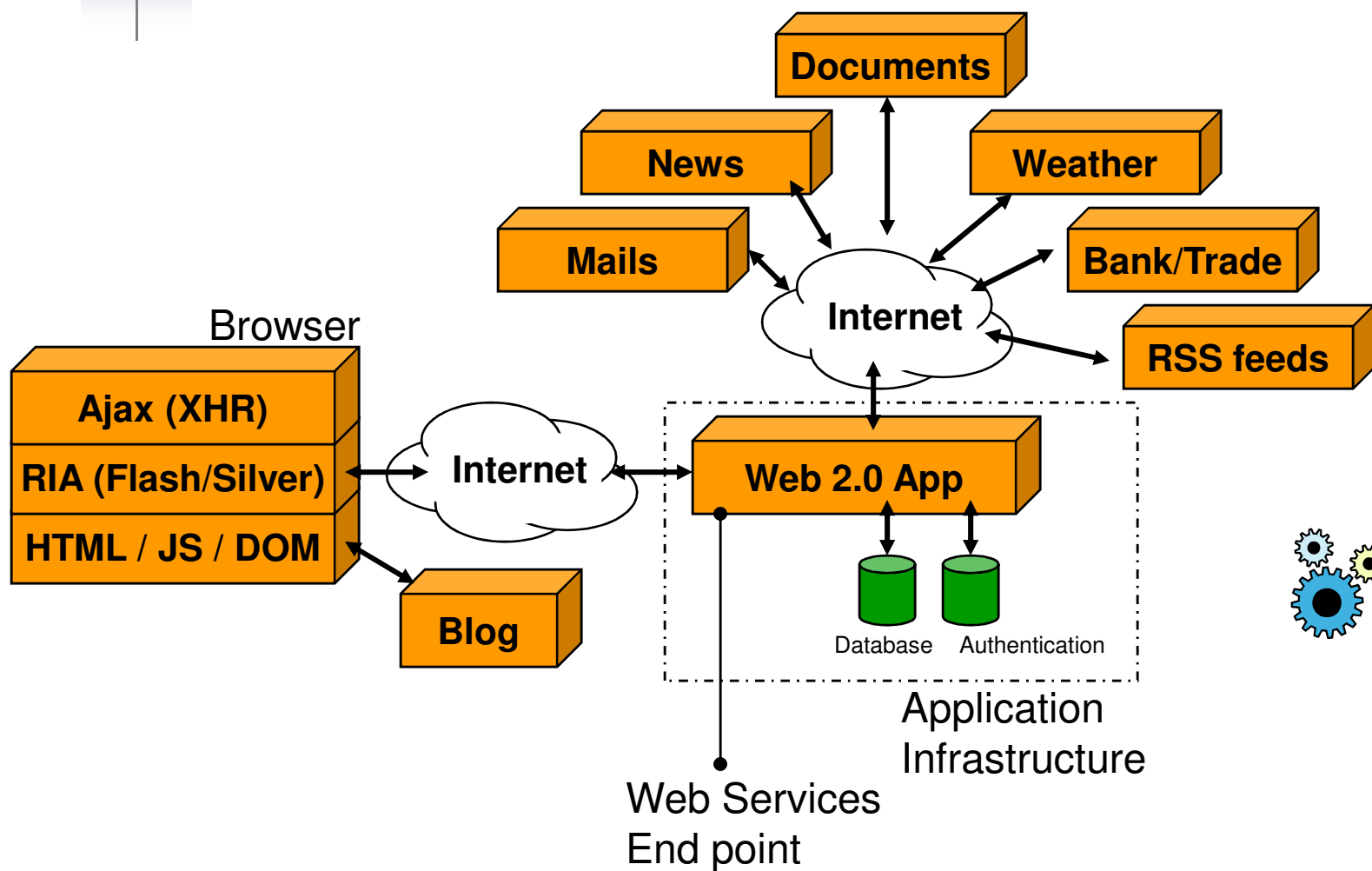
AppSec dynamics

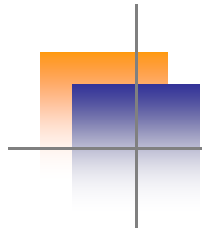
New Top Ten 2004
A1 Unvalidated Input
A2 Broken Access Control
A3 Broken Authentication and Session Management
A4 Cross Site Scripting (XSS) Flaws
A5 Buffer Overflows
A6 Injection Flaws
A7 Improper Error Handling
A8 Insecure Storage
A9 Denial of Service
A10 Insecure Configuration Management

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A10 – Failure to Restrict URL Access	A7 – Failure to Restrict URL Access
<not in T10 2007>	A8 – Unvalidated Redirects and Forwards (NEW)
A8 – Insecure Cryptographic Storage	A9 – Insecure Cryptographic Storage
A9 – Insecure Communications	A10 – Insufficient Transport Layer Protection
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

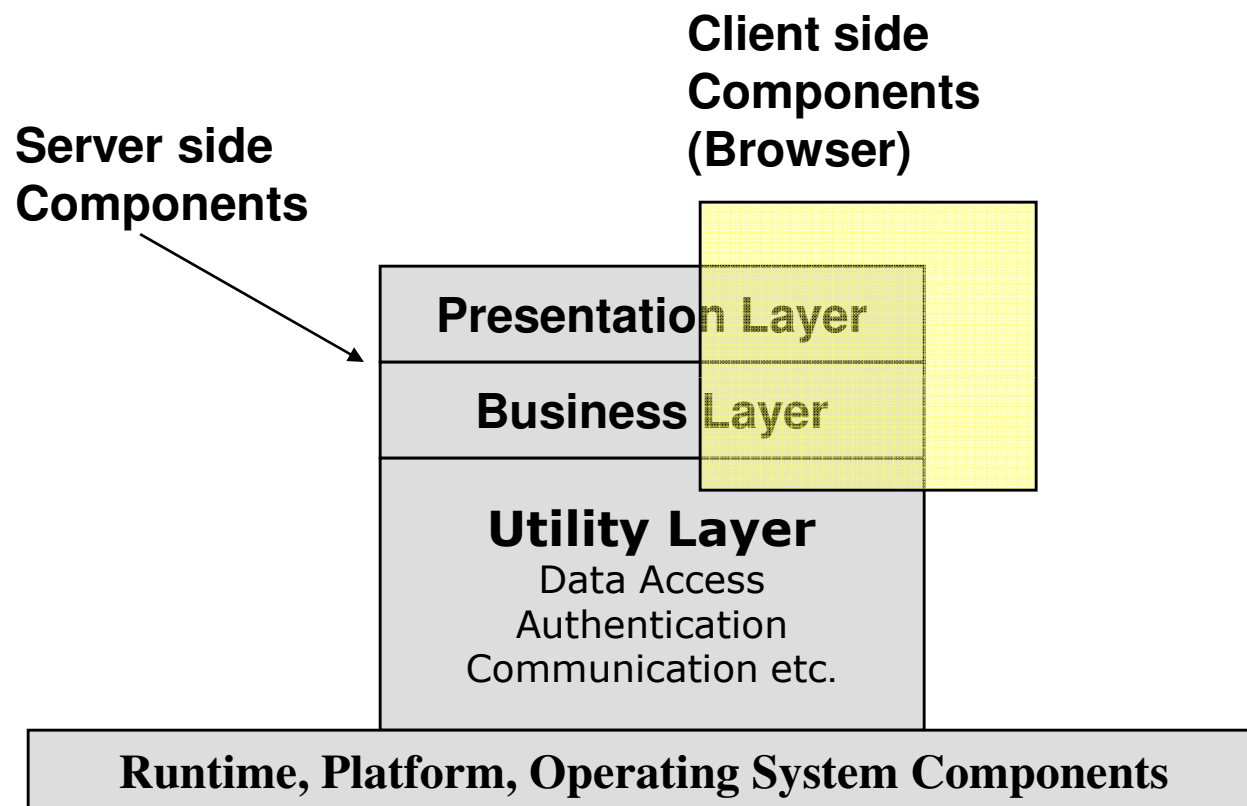


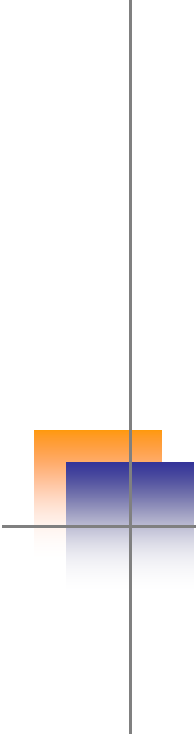
Next Generation Architecture



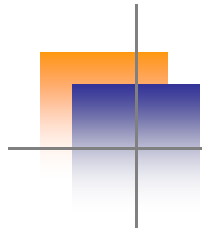


Breaking traditional layers





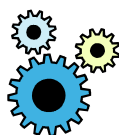
Why Reverse Engineering?



Scan vs. Reversing

- *Scope of coverage*

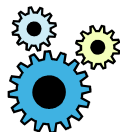
- Scanning method uses crawling and spidering to determine all possible resources

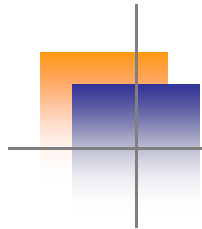


- Application assets are residing in Ajax, Flash and Silverlight, it makes asset detection very difficult and scanning approach fails in many cases.

- *Discovery and Detection*

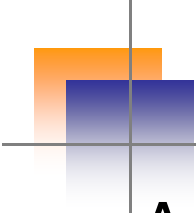
- Scanning uses signature analysis for vulnerability detection. Example, it looks for ODBC error for SQL injection and so on, fails with AMF or other methods.





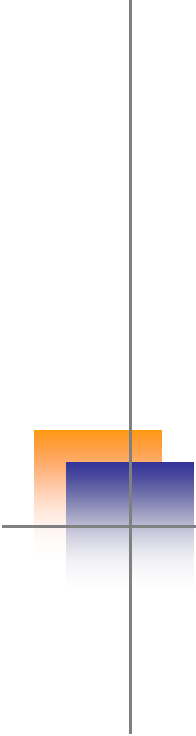
Scan vs. Reversing

- *Accuracy of Vulnerability*
 - Accuracy of vulnerability is very important as well.
 - Scan is inaccurate with RIA
 - Came up false +/-
- *Cause Identification*
 - One of the major challenges is to identify actual cause of the vulnerability.
 - Scan shows symptoms
 - Reversing may pin point the cause

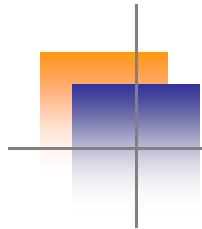


Challenges

- Application entry points are scattered and multiple, number of entry points are coming over HTTP traffic and some times tricky to detect.
- Web 2.0 applications are running with Web Services using protocols like SOAP, XML-RPC or REST.
- Application layer tracing is also difficult and challenging since it is across multiple pages along with some intermediate framework code.
- Source code analysis has different technologies and modules involved in the process and it makes very difficult.
- One important aspect is client side coding and modules like Ajax, JavaScript, Flash and Silverlight.



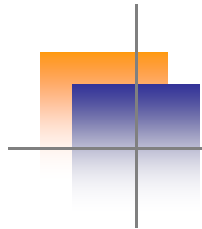
Attacks and Reverse Engineering



Client side technologies

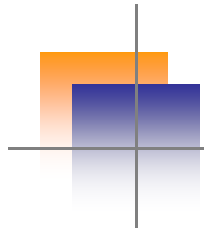
- Ajax
- Flash/Flex
- Silverlight

- There are several issues from development standpoint and it is imperative to identify.



Vulnerabilities

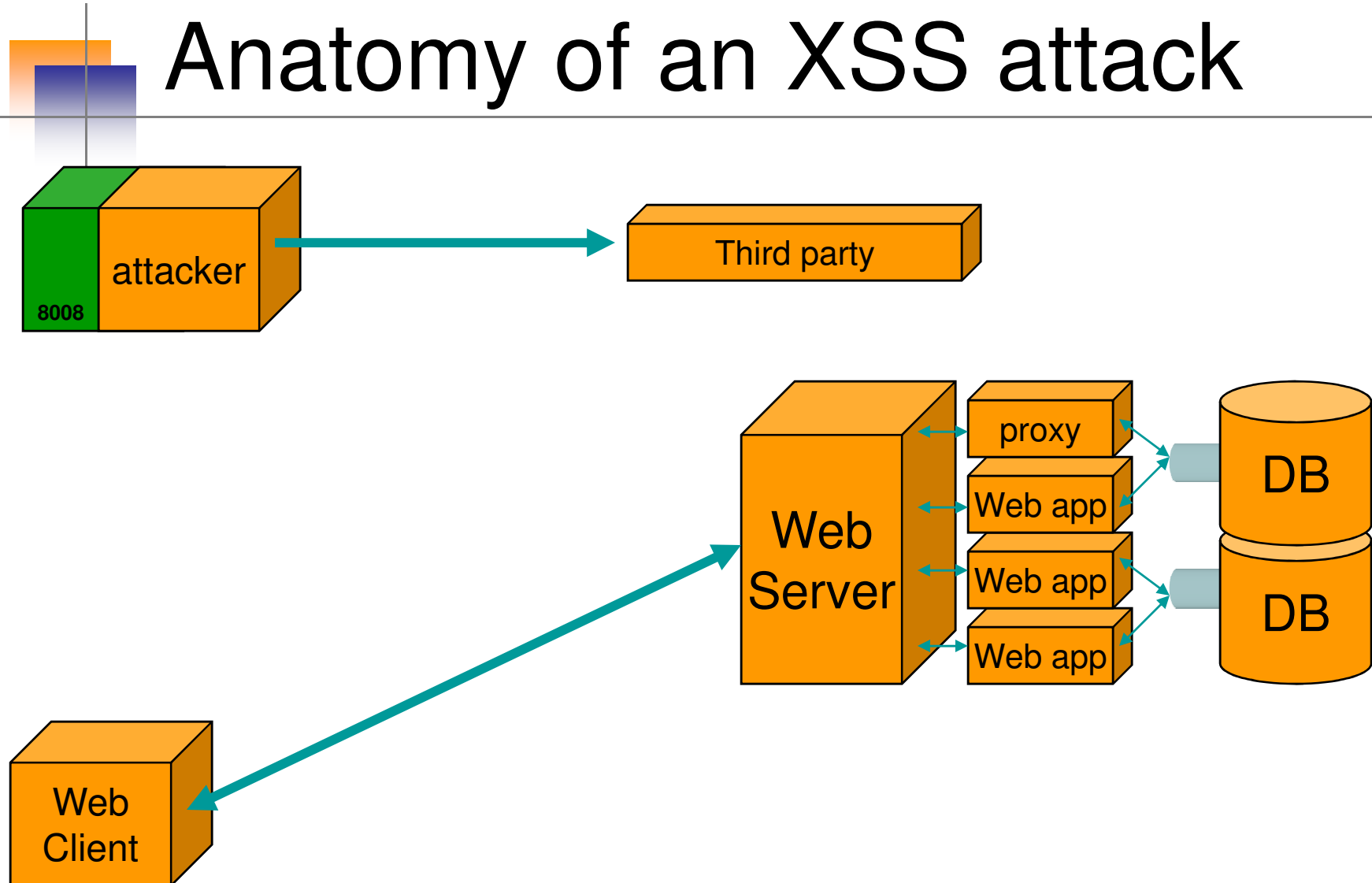
- DOM based XSS with Ajax
- Cross Site Flashing and Flash based XSS
- Business logic residing in client side
- Information leakage from client side code
- Discovering hidden paths and predictable resources



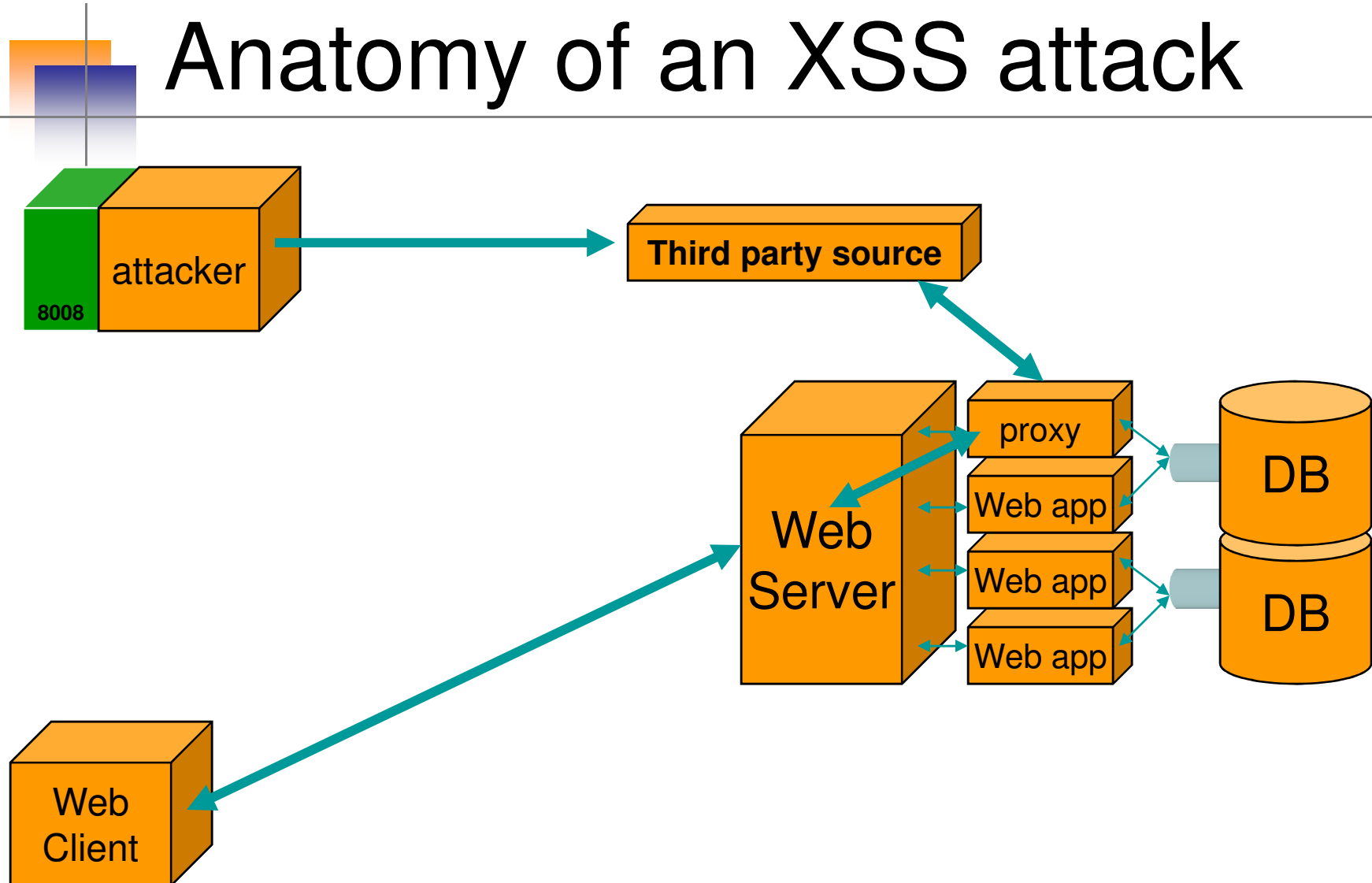
DOM based XSS

- Ajax based XSS is relatively new way of attacking the client
- Code written on browser end can be vulnerable to this attacks
- Various different structures can have their own confusion
- Information processing from un-trusted sources can lead to XSS

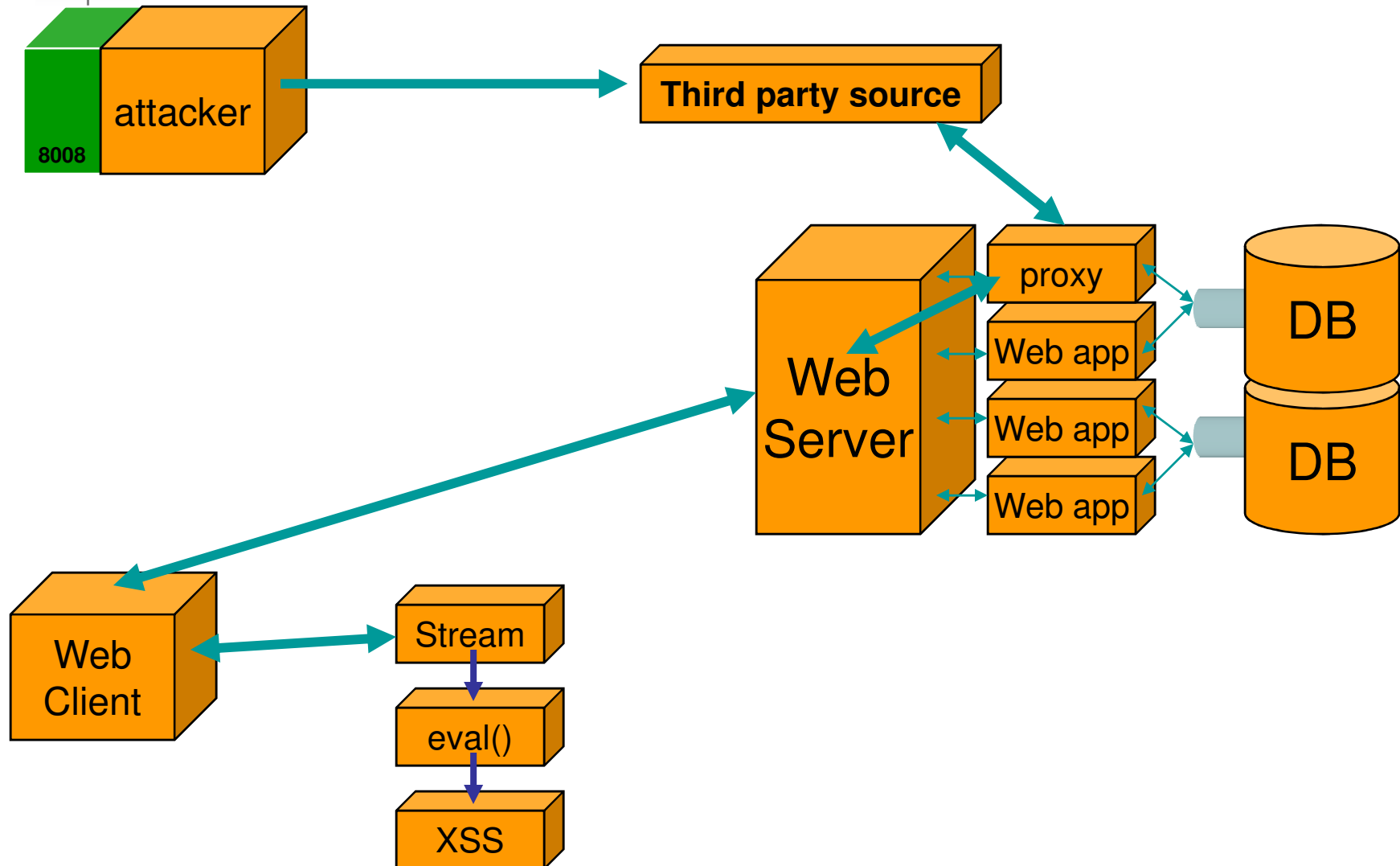
Anatomy of an XSS attack

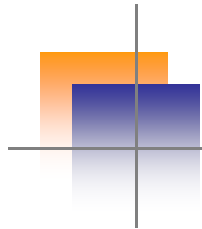


Anatomy of an XSS attack



Anatomy of an XSS attack

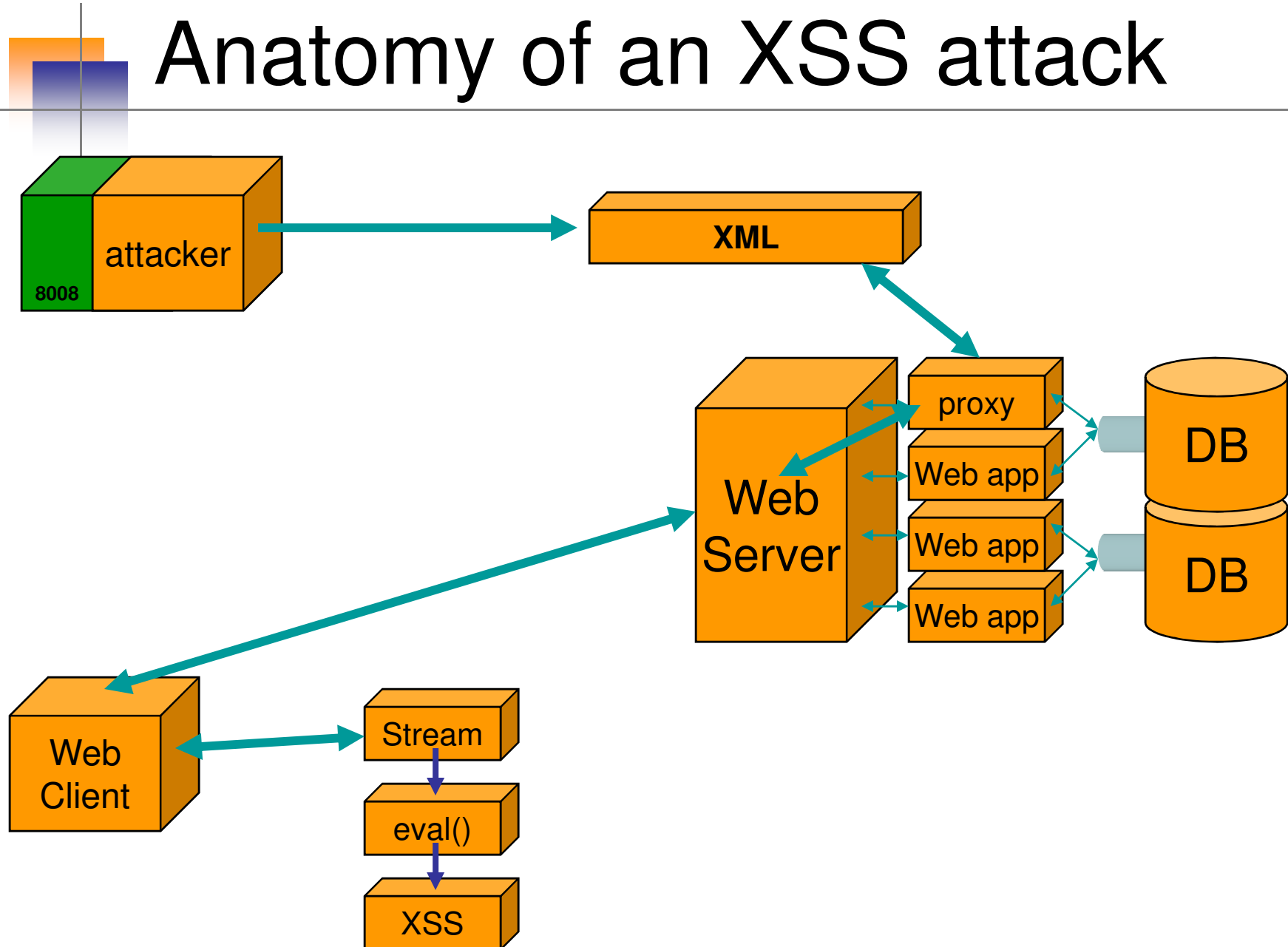




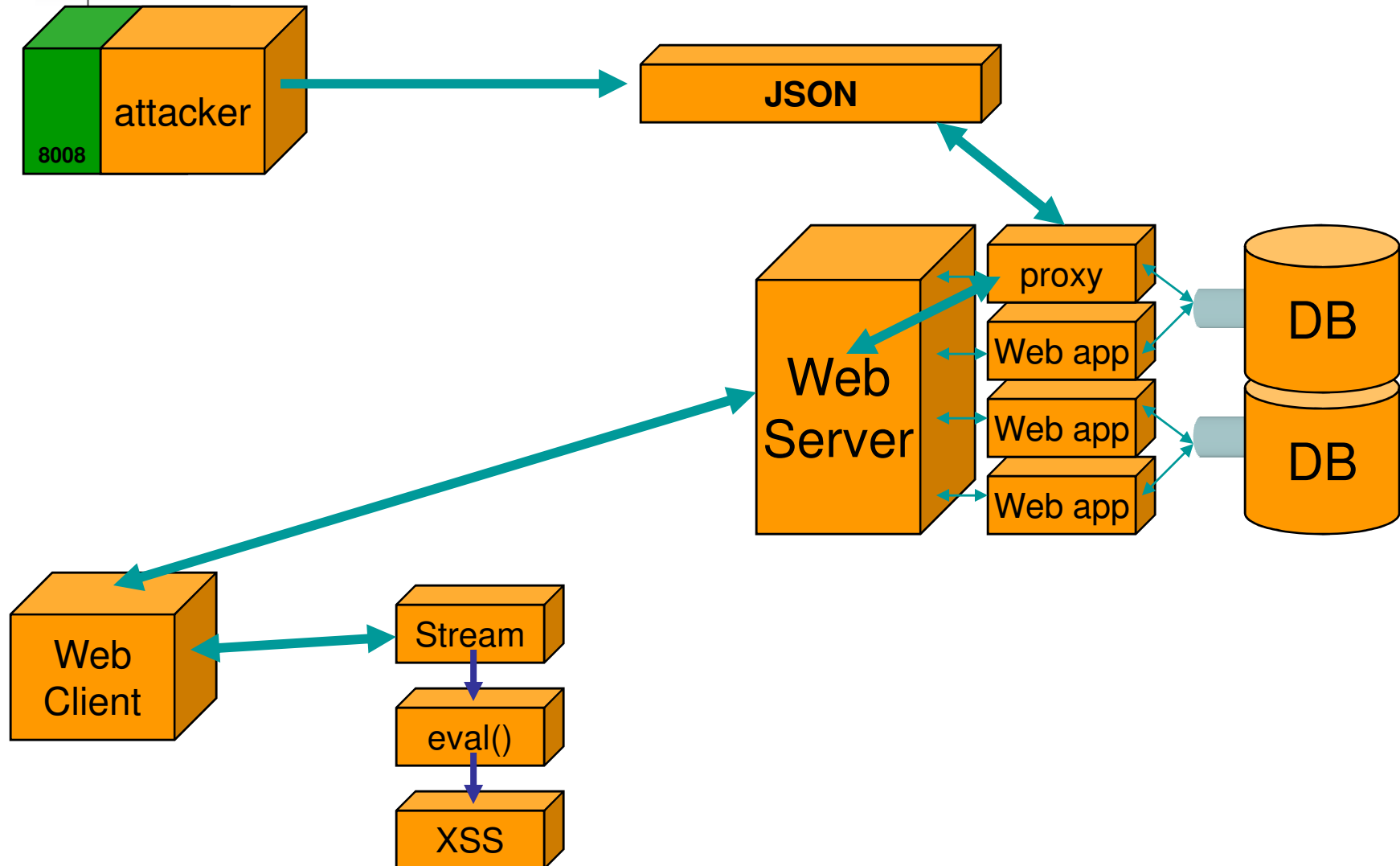
DOM based XSS

```
if (http.readyState == 4) {  
    var response = http.responseText;  
    var p = eval("(" + response + ")");  
    document.open();  
    document.write(p.firstName+"<br>");  
    document.write(p.lastName+"<br>");  
    document.write(p.phoneNumbers[0]);  
    document.close();  
}
```

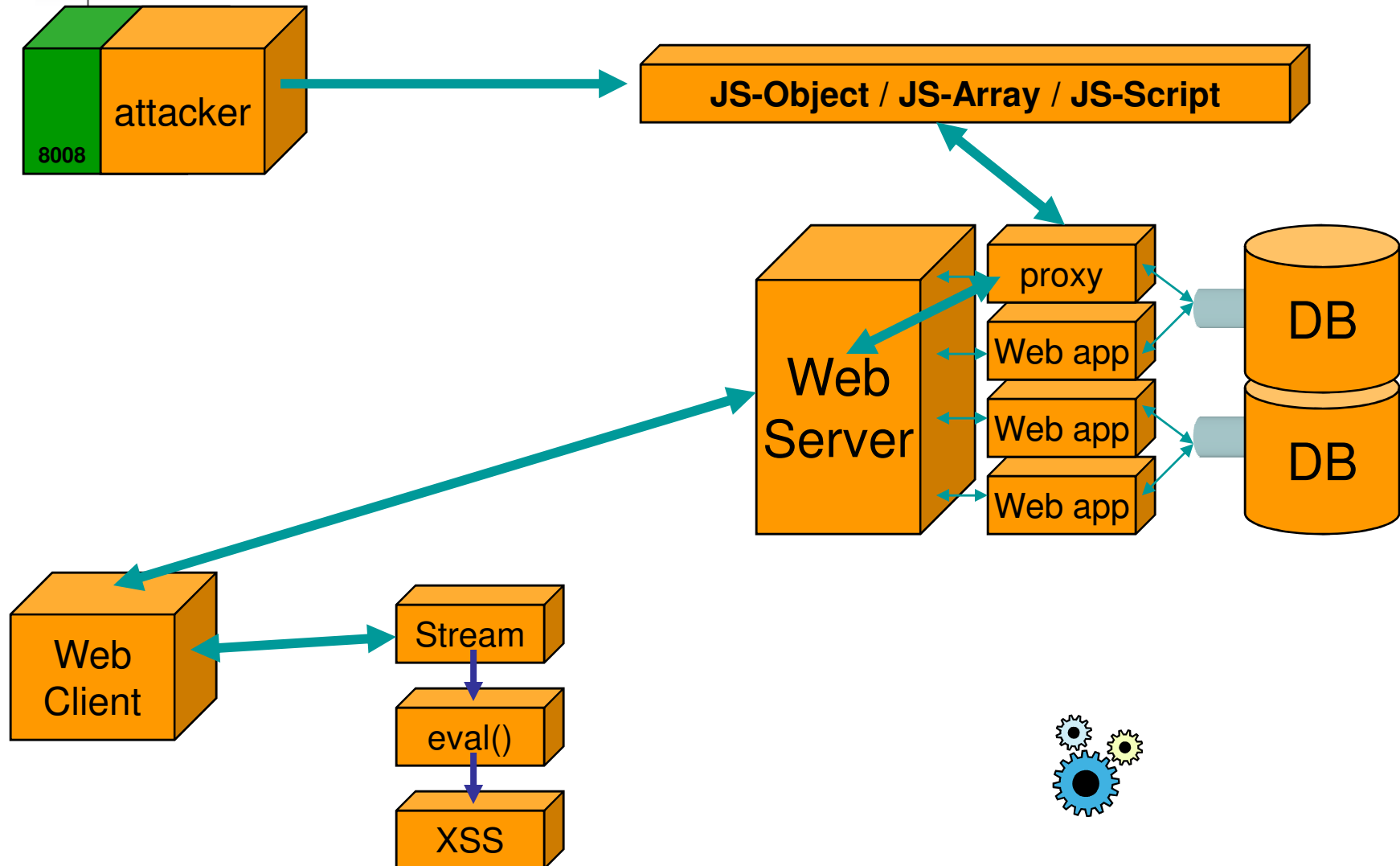
Anatomy of an XSS attack

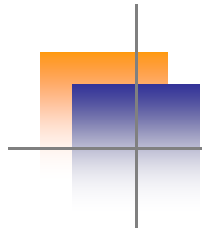


Anatomy of an XSS attack



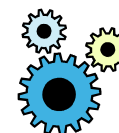
Anatomy of an XSS attack

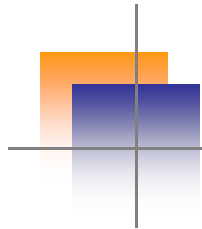




DOM based XSS

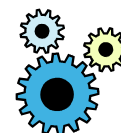
document.write(...)
document.writeln(...)
document.body.innerHTML=...
document.forms[0].action=...
document.attachEvent(...)
document.create...(...)
document.execCommand(...)
document.body. ...
window.attachEvent(...)
document.location=...
document.location.hostname=...
document.location.replace(...)
document.location.assign(...)
document.URL=...
window.navigate(...)

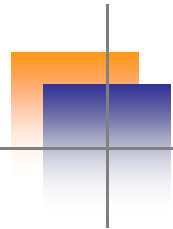




Ajax – Reverse Eng.

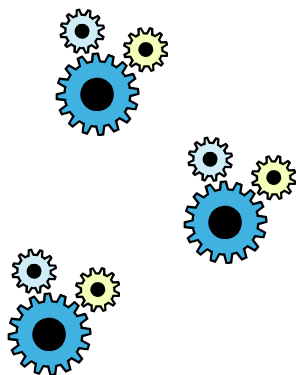
- Looking into JavaScript functions
- Looking for calls
- Looking for sources and trust
- Debugging JavaScript
- Tools and scan ...



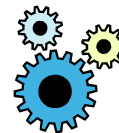


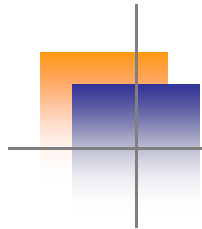
Ajax components

- RSS feeds
- Mashups
- Widgets



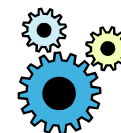
- Possible to perform XSS or Clickjacking.
- Inter-widget spying is also reality
- Reverse engineering can help in identifying it.

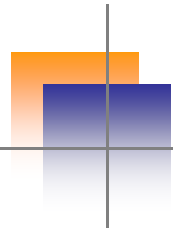




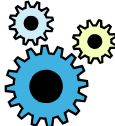
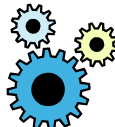
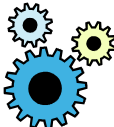
Debugging logic

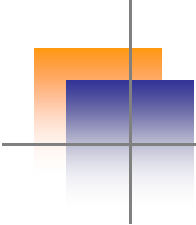
- It is possible to reverse engineer business logic
- Critical business layer information on the client's stack
- Manipulating calls and checking it out
- Debugging through firebug



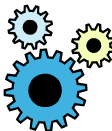


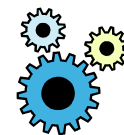
Flash based RIA

- XSS with flash 
- Usage of calls 
- Global access
- “asfunction” calls and methods
- Cross site flashing 
- Exploiting redirecting methods



Dissecting RIA

- SWF decompiler 
- Analyzing action scripts and other files 
- Tools can help in fetching information
- Identifying back-end calls
- AMF stream and fuzzing them
- Silverlight de-compilation on similar lines 
- Lot of analysis and findings on these lines





Conclusion – Questions?
